Efficient Loss Function by Minimizing the Detrimental Effect of Floating-point Errors on Gradient-based Attacks

Yunrui Yu
Department of Computer Science
State Key Lab of IOTSC
University of Macau, Macau SAR, China.
yb97445@um.edu.mo

Cheng-Zhong Xu*
Department of Computer Science
State Key Lab of IOTSC
University of Macau, Macau SAR, China.
czxu@um.edu.mo

Abstract

Attackers can deceive neural networks by adding human imperceptive perturbations to their input data; this reveals the vulnerability and weak robustness of current deep-learning networks. Many attack techniques have been proposed to evaluate the model's robustness. Gradient-based attacks suffer from severely overestimating the robustness. This paper identifies that the relative error in calculated gradients caused by floating-point errors, including floating-point underflow and rounding errors, is a fundamental reason why gradient-based attacks fail to accurately assess the model's robustness. Although it is hard to eliminate the relative error in the gradients, we can control its effect on the gradient-based attacks. Correspondingly, we propose an efficient loss function by minimizing the detrimental impact of the floating-point errors on the attacks. Experimental results show that it is more efficient and reliable than other loss functions when examined across a wide range of defence mechanisms.

1. Introduction

AI with deep neural networks (DNNs) as the core [24] has achieved great success in different research directions and has been widely used in many safety-critical systems, such as aviation [1, 8, 35], medical diagnosis [27, 49], self-driving [4, 26, 47] , etc. However, a severe problem with current DNNs is that they are vulnerable to adversarial attacks. Adding human imperceptive perturbations to input data can mislead the model to output incorrect results [16, 45]. Such vulnerability poses a significant security risk to all DNN-based systems. Therefore, increasing the models' robustness and providing efficient and reliable evaluation methods are becoming increasingly urgent and vital.

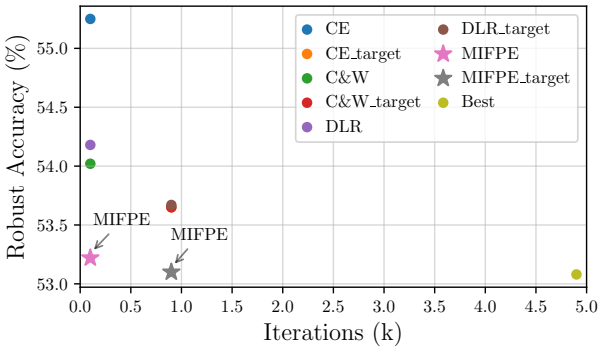


Figure 1. Comparison of the effectiveness of non-targeted and multi-targeted PGD 100 iterations attacks using various loss functions on the CIFAR-10 dataset. The best results were obtained from RobustBench[1] using an ensemble attack with a minimum of 4900 iterations. The defence model is from [59].

Numerous defence strategies [6, 10, 13, 16, 29, 31, 55] to improve the model's robustness and many attacks [2, 29, 33, 41, 50, 52, 54, 59] to evaluate these strategies have been proposed. Currently, the most efficient attack method is known as white-box attack, where the attacker has complete knowledge of the target defending strategy, including its model architecture, parameters, detail of the training algorithm, dataset, etc. A typical example is Projective gradient descent (PGD) [29] for evaluating the model's robustness. However, studies have shown that PGD with cross-entropy (CE) loss fails to provide accurate evaluation results [6, 41] and significantly overestimates the model's robustness(Figure 1). To enhance the evaluation accuracy, recent powerful evaluation methods were proposed to ensemble diverse attack algorithms [10, 30]. However, their performance is at the cost of high computational overhead.

The research community attributes the failure of gradient-based attacks to gradient masking [15]. The core of gradient masking is that the calculated gradients are not

*Corresponding author.
[1] https://robustbench.github.io/

necessarily efficient in guiding the generation of adversarial examples. The causes of inefficient gradients may be related to floating-point errors or other factors. This paper reveals that the relative error in the gradients caused by floating-point errors is one of the fundamental reasons for the failure of gradient-based attacks, and the relative error is directly affected by the value of the difference of the first and second largest in logits(DFSL). Although it is hard to eliminate the relative error in the gradients caused by floating-point errors completely, we can control its effect on the gradient-based attacks. Thus, we propose a new efficient loss function by *minimizing the impact of floating-point errors* (MIFPE) on gradient-based attacks. To summarize, our main contributions are as follows:

- We provide a comprehensive demonstration of how the presence of floating-point errors leads to the failure of gradient-based attacks due to the relative error in calculated gradients.
- We show that the floating-point rounding errors may lead to attack's failure due to growing relative errors as samples reach the classification boundary.
- We demonstrate that when the model's robustness is overestimated primarily due to floating-point errors, the loss function discarding partial elements of logits causes performance degradation in the gradient-based attack.
- We show that minimizing the impact of floating-point errors can improve the efficiency and accuracy of gradient-based attacks.
- Empirical results demonstrate that MIFPE tends to outperform the competing loss functions in terms of attack performance and computational efficiency.

2. Preliminaries & Related Work

We define the classification model as $f_{\boldsymbol{\theta}}(\mathbf{x})$, where $\boldsymbol{\theta}$ represents the model parameters. $f_{\boldsymbol{\theta}} : \mathbf{x} \subset \mathbb{R}^{C\times H\times W} \rightarrow \mathbf{z} \subset \mathbb{R}^k$ is a classifier that maps input data $\mathbf{x}$ to output logits $\mathbf{z}$, where C, H, and W are parameters that describe the input data's channel number, height, and width, respectively. Additionally, K represents the number of classes from the model's output. We define the label of the input data as y.

Attackers aim to breach defence strategies by generating adversarial examples $\hat{\mathbf{x}}$, which can maximize the loss function $L\left(f_{\boldsymbol{\theta}}\left(\hat{\mathbf{x}}\right), y\right)$ and deceive the model into producing incorrect results, $\arg\max f_{\boldsymbol{\theta}}(\hat{\mathbf{x}}) \neq y$. Achieving this requires the attackers to find imperceptible perturbations $\delta = \hat{\mathbf{x}} - \mathbf{x}$ that satisfy the constraint $\|\delta\|p \leq \varepsilon$, where ε is the magnitude and p is the norm of the perturbation. Various perturbation options have been explored in literature, including the one-pixel attack [44, 44], l_2 attacks [6, 31, 45], and $l\infty$ attacks [16, 29]. The attackers can find an adversarial example $\hat{\mathbf{x}} \in \mathcal{I}$ by maximizing the loss. This process can be formulated as :

$$\hat{\mathbf{x}} = \mathbf{x} + \max_{\|\delta\|_p \in \varepsilon} L\left(f_{\boldsymbol{\theta}}\left(\mathbf{x}+\delta\right), y\right), \tag{1}$$

The white-box scenario is the most rigorous test for defence strategies, as attackers have complete knowledge about the models' architecture, parameters, training data, and algorithm implementation details. This is a highly challenging situation for defence strategies. Attackers can exploit nearly all vulnerabilities in the defence strategies and perform tailored attacks. If defence strategies remain robust against white-box attacks, they can ensure the system's security even in the worst-case scenario. In practice, the gradient-based method is commonly used to solve (1).

Many gradient-based white-box attacks methods have been proposed, which include fast gradient-sign method (FGSM) [16], basic iterative method (BIM) [23], projected gradient descent (PGD) [29], momentum iterative method (MIM) [13] and fast adaptive boundary attack (FAB) [9] for ℓ_∞-norm attacks. Carlini and Wagner (C&W) [6] and DeepFool [31] can additionally carry out ℓ_2-norm attacks. PGD is one of the widely used methods to evaluate the model's robustness in the white box scenario, which finds an adversarial example by performing the following iterative update:

$$\hat{\mathbf{x}}_{i+1} = \mathcal{P}_{\epsilon,\mathbf{x}}\left(\hat{\mathbf{x}}_i + \alpha_i \operatorname{sign}\left(\nabla_{\hat{\mathbf{x}}_i} \mathcal{L}^{\text{ce}}\left(f_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_i), y\right)\right)\right), \tag{2}$$

Initially, $\hat{\mathbf{x}}_0 = \mathcal{P}_{\epsilon,\mathbf{x}}(\mathbf{x}+\mathbf{u})$, where $\mathbf{u} \sim \mathcal{U}\left([-\epsilon, \epsilon]\right)$, *i.e.* $\mathbf{u}$ is a uniform random noise bounded by $[-\epsilon, \epsilon]$. The function $\mathcal{P}_{\epsilon,\mathbf{x}}$: $\mathbb{R}^{C\times H\times W} \rightarrow \mathcal{I}$ first clips the range of its input into the ϵ-ball ℓ_p-distance neighbour of the original image $\mathbf{x}$ and then clips its input into the range of $\mathcal{I}$. The term $\nabla_{\hat{\mathbf{x}}_i} \mathcal{L}^{\text{ce}}\left(f_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_i), y\right)$ computes the gradient of the loss w.r.t. the input $\hat{\mathbf{x}}_i$. Finally, α_i is the step size, and for each element in the tensor $\boldsymbol{g}$, $\operatorname{sign}(\boldsymbol{g})$ returns one of 1, 0 or -1, if the value is positive, zero or negative, respectively.

CE is a commonly used loss function for training deep neural network models and evaluating their robustness in classification tasks. However, the presence of exponential operations in CE makes it vulnerable to underflow and rounding errors with limited floating-point precision. Gradient-based attacks with the CE loss may experience vanishing/inefficient gradients and difficulty converging while suffering from a severe overestimation of the model's robustness. [6, 10]. Recent attack methods use *surrogate losses* for gradient computation [6, 18] and optimize an alternative target by replacing $\mathcal{L}^{\text{ce}}$ with a custom surrogate loss function. $\mathbf{z} = f_{\boldsymbol{\theta}}\left(\hat{\mathbf{x}}\right)$ is output logits of the model, and $\mathbf{z}_i$ is the i^{th} component of $\mathbf{z}$, two typical examples to replace the CE objective include the difference-of-logits (DL) [6], also known as hinge loss, where $i \in \{1, 2, ..., K\}$

and $i \neq y$:

$$\mathcal{L}^{\text{cw}}(\mathbf{z},\mathrm{y}) = -\mathbf{z}_y + \max_{i \neq y} \mathbf{z}_i, \tag{3}$$

and the difference-of-logits ratio (DLR) loss [10]:

$$\mathcal{L}^{\text{dlr}}(\mathbf{z},\mathrm{y}) = \frac{-\mathbf{z}_y + \max_{i \neq y} \mathbf{z}_i}{\mathbf{z}_{\pi 1} - \mathbf{z}_{\pi 3}}, \tag{4}$$

where $\mathbf{z}_{\pi_1}$ and $\mathbf{z}_{\pi_3}$ respectively denote the 1^{th} and 3^{th} largest components of $\mathbf{z}$. It is clear that maximizing the alternative goals is consistent with the original objective of making $\arg\max f(\hat{\mathbf{x}}) \neq y$. In addition, since surrogate losses avoid exponential operations, they are not prone to underflow, allowing gradient-based optimization to escape suffering from the overestimation caused by the floating-point underflow errors. However, the floating-point underflow errors are not the sole reason for all overestimations related to floating-point errors. At the same time, the new loss function discards some elements of $\mathbf{z}$, reducing the effectiveness of the attack and causing gradient-based attacks with surrogate losses to overestimate the model's robustness.

Many auxiliary techniques can assist existing attack methods to improve their performance. For example, momentum-based updates can improve the attack's convergence rate and are widely used by white-box attacks [13, 18]. A step schedule with decreasing step size based on the number of iterations can improve the overall success rate [10, 18, 56]. Multi-target attacks can outperform non-target attacks in the evaluation results at the cost of more iterations since they require enumerating multiple most likely target labels [18, 33, 34]. In addition to the above strategies, multiple restarts can also improve the effectiveness of gradient-based attacks by adding different random initial perturbations to the input data to diversify the starting point of the attack and reduce the impact of suboptimal solutions [5, 46]. Finally, several recent publications have shown that leveraging latent features in attacks can enhance the transferability of black-box attacks [21, 22] and the performance of white-box attacks [56].

However, even with so many auxiliary strategies mentioned above, using any single attack is still challenging to accurately evaluate the model's robustness. Attackers start using an ensemble of multiple attack strategies to improve the attack's performance by introducing diversity [5, 10, 28]. They often suffer from high computational costs. Even though these ensemble strategies are more powerful than any single attack, they still fail to provide an assessment of the ensemble defence strategy's robustness accurately [48, 57].

3. Efficient Loss Function

Assuming that the input $\hat{\mathbf{x}}$ has the correct label y, we compute $\mathbf{z} = f_{\boldsymbol{\theta}}(\hat{\mathbf{x}})$, where $\boldsymbol{\theta}$ represents the model parameters. Next, we sort the values in $\mathbf{z}$ in a descending order, where $\mathbf{z}_{\pi 1}$ is the maximum value. We define Δ as the difference between the maximum and the second maximum value, i.e., $\Delta = \mathbf{z}_{\pi 1} - \mathbf{z}_{\pi 2} \geq 0$. The cross-entropy loss at $\mathbf{z}$ becomes

$$CE(\mathbf{z},y) = -\log p_y = -\log \frac{e^{\mathbf{z}_y - \mathbf{z}_{\pi 1}}}{\sum_{i=1}^{K} e^{\mathbf{z}_i - \mathbf{z}_{\pi 1}}}, \tag{5}$$

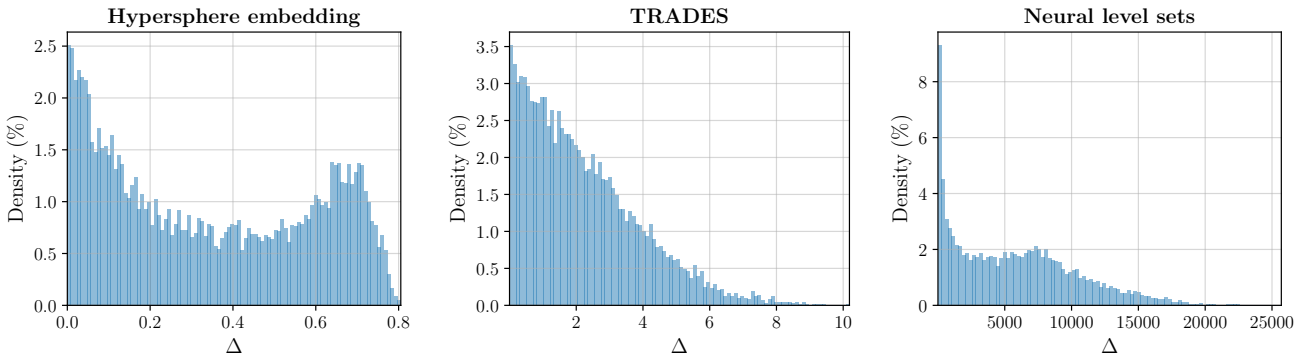
$$\begin{aligned} \nabla_{\hat{\mathbf{x}}} CE(\mathbf{z},y) = (-1 + p_y)\, \nabla_{\hat{\mathbf{x}}} (\mathbf{z}_y - \mathbf{z}_{\pi 1}) \\ + \sum_{i \neq y} p_i \nabla_{\hat{\mathbf{x}}} (\mathbf{z}_i - \mathbf{z}_{\pi 1}), \end{aligned} \tag{6}$$

where $p_i = e^{\mathbf{z}_i - \mathbf{z}_{\pi 1}} / \sum_{j=1}^{K} e^{\mathbf{z}_j - \mathbf{z}_{\pi 1}}$, $i \in \{1, 2, 3, ..., K\}$.

Exponential operations in Equation (5), specifically $e^{\mathbf{z}_{\pi i} - \mathbf{z}_{\pi 1}}, i \in \{2, ..., K\}$, where $\mathbf{z}_{\pi i} - \mathbf{z}_{\pi 1} \leq \mathbf{z}_{\pi 2} - \mathbf{z}_{\pi 1} = -\Delta \leq 0$, can cause floating-point underflow errors in the CE loss gradient. To investigate this issue, we analyze the properties of $e^{-\Delta}$ and observe that when Δ surpasses a threshold value denoted by λ, $e^{-\Delta}$ underflows and leads to significant inaccuracies in the gradient calculation. Specifically, when $\Delta \geq \lambda$ and $\mathbf{z}_y = \mathbf{z}_{\pi 1}$, the predicted probability $p_y = 1$ and the calculated gradient $\nabla_{\hat{\mathbf{x}}} CE(\mathbf{z},y) = 0$. The value of λ varies with different floating-point arithmetic precisions, with approximate values of 16.64, 103.28, and 744.44 for half-precision, single-precision, and double-precision, respectively.

Based on the findings, we identify that the root cause of the calculated gradient being equal to zero is the result of Δ exceeding λ. To further investigate the relationship between the value of Δ and the overestimation of the model's robustness, we analyze the distribution of Δ values in defence models that tend to be overestimated by gradient-based attacks under single-precision. The results in Figure 2 indicate that the range of Δ values varies significantly among different defence models. Notably, the values of Δ in 2a and 2b do not exceed 10, which is significantly smaller than $\lambda \approx 103.28$ for single-precision. As a result, these models do not suffer from floating-point underflow errors. In contrast, most of the Δ values in 2c exceed the threshold that leads to floating-point underflow errors. It is evident that not all models suffer from floating-point underflow errors, and other reasons associated with floating-point errors cause gradient-based attacks to fail. This is because in addition to floating-point underflow errors, floating-point errors have another component: floating-point rounding errors when $\Delta < \lambda$. These errors result in a non-negligible relative error in $\nabla_{\hat{\mathbf{x}}} CE(\mathbf{z},y)$. We define the relative error as $\delta_{CE} = \delta(\nabla_{\hat{\mathbf{x}}} CE(\mathbf{z},y))$.

We know that $\delta_{CE} = 100\%$ when $\Delta \geq \lambda$. We need to further analyze the change pattern of δ_{CE} when $\Delta < \lambda$. A multi-iteration gradient-based attack involves adding perturbations to $\hat{\mathbf{x}}$ at each iteration, leading to constant changes in $\mathbf{z}$, Δ, and $\nabla_{\hat{\mathbf{x}}}(\mathbf{z}_{\pi i} - \mathbf{z}_{\pi 1})$ for $i \in 1, 2, ..., K$ after each



(a) The Δ distribution of [33]

(b) The Δ distribution of [59]

(c) The Δ distribution of [3]

Figure 2. The distributions of $\Delta = \mathbf{z}_{\pi 1} - \mathbf{z}_{\pi 2}$ for the defending models on the CIFAR-10 dataset are averaged across 100 bins.

modification. This makes it difficult to analyze how δ_{CE} changes when $\Delta < \lambda$. However, we can indirectly examine the effect of floating-point errors on the attack's performance by analyzing the impact of such errors on a single step. To do so, we introduce a scaling factor c and define the scaled value of Δ as $T = c\Delta$. By varying c while keeping $\hat{\mathbf{x}}$, $\mathbf{z}$, and $\nabla_{\hat{\mathbf{x}}}(\mathbf{z}_{\pi i} - \mathbf{z}_{\pi 1})$ fixed during gradient calculation, we can investigate the relationship between δ_{CE} and T for different values of T.

To compute the scale factor, we need to detach the gradient information from Δ since $\mathbf{z}$ contains the gradient information during the actual program running. Thus, we define the scale factor as $c = T/\Delta_{\text{detach}}$. The use of the detach operation in the calculation of c removes the gradient information from the Δ_{detach}, making it distinct from the Δ that retains gradient information. When $\mathbf{z}_y \neq \mathbf{z}_{\pi 1}$, the model output is already incorrect. Therefore, we only focus on the relationship between δ_{CE} and T when $\mathbf{z}_y = \mathbf{z}_{\pi 1}$, at which point $\nabla_{\hat{\mathbf{z}}} CE\left(\mathbf{cz}, y\right) = c\sum_{i \neq y} p_i^c \nabla_{\hat{\mathbf{x}}}\left(\mathbf{z}_i - \mathbf{z}_{\pi 1}\right)$, where $p_i^c = e^{c(\mathbf{z}_i - \mathbf{z}_{\pi 1})} / \sum_{j=1}^{K} e^{c(\mathbf{z}_j - \mathbf{z}_{\pi 1})}$.

To get a more intuitive picture of the relationship between δ_{CE} and T, we plotted the figure of δ_{CE} on the binary classification task with $K = 2$, At this point, $\nabla_{\hat{\mathbf{z}}} CE\left(\mathbf{cz}, y\right) = c\sum_{i \neq y} p_i^c \nabla_{\hat{\mathbf{x}}}\left(\mathbf{z}_i - \mathbf{z}_{\pi 1}\right) = \mathrm{c}p_2^c \nabla_{\hat{\mathbf{x}}}\left(\mathbf{z}_{\pi 2} - \mathbf{z}_{\pi 1}\right) \propto \mathrm{c}p_2^c$. Therefore, $\delta_{CE} \propto \delta\left(\mathrm{c}p_2^c\right)$. In Figure 3, we can see that for the same $\hat{\mathbf{x}}$,when $\mathbf{z}$ and $\nabla_{\hat{\mathbf{x}}}\left(\mathbf{z}_i - \mathbf{z}_{\pi 1}\right)$, $i \in \{1, 2, ..., K\}$ are all fixed, δ_{CE} varies with T. When $T \geq \lambda$, $\delta_{CE} = 100\%$ In contrast, When $T < \lambda$, The relative error δ_{CE} caused by the floating-point rounding errors is much more minor, varies continuously with T, and reaches the minimum when T is approximately 1. But one thing that needs to be highlighted is that after the relative error reaches the minimum, it starts to increase as T gets closer to 0.

Following the same operation, we add a scale factor c to Δ and hold $T = c\Delta$ constant during each iteration of the multi-iteration attack. This technique, which we refer to as FT-PGD, allows us to assess the model's robustness with a fixed T value. In Figure 4, we evaluate the model's robustness using FT-PGD at various T values. By comparing the relative floating-point error in Figure 3 with the attack's performance in Figure 4, we observe a strong correlation between the relative error δ_{CE} and the attack's effectiveness. Thus, the relative error is a critical factor that contributes to overestimating the model's robustness. Furthermore, we identify two distinct phases in the impact of floating-point errors on gradient-based attacks due to the CE loss, which depend on the relationship between T and λ.

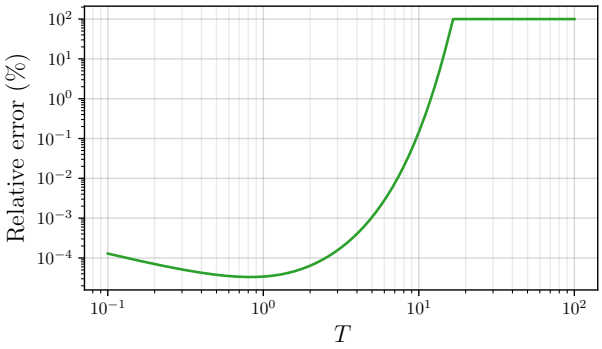


Figure 3. The relative error of the gradient information computed using half-precision floating-point operations on a binary classification task with the CE loss varies with T for a given adversarial example within a single iteration.

(a) *The attack failed due to floating-point underflow errors.* When $T \geq \lambda$, floating-point underflow causes the calculated gradient equal to zero and $\delta_{CE} = 100\%$. As a result, the input perturbation becomes zero, and the model's robustness is measured to be equal to its clean accuracy, which leads to a significant overestimation of the model's robustness.

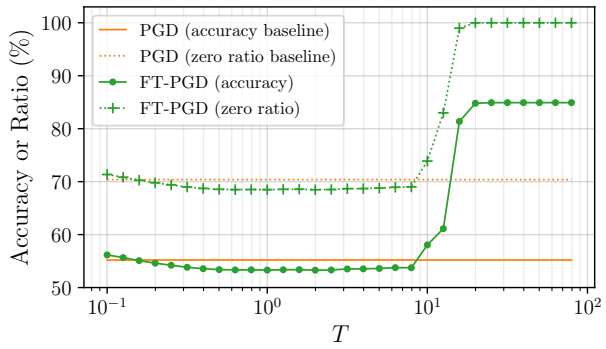


Figure 4. A higher zero ratio, which represents the proportion of zero elements in the perturbation, leads to larger overestimations when evaluating model robustness. We assess model robustness using 100 iterations of FT-PGD with CE loss on the CIFAR-10 dataset under half-precision floating-point arithmetic. The model is obtained from [59].

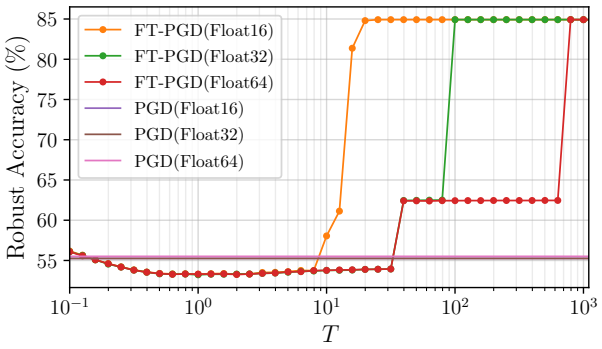


Figure 5. The model's robustness test results on the CIFAR-10 dataset under Float16, Float32, and Float64 follow a similar pattern of variation, except that the thresholds that lead to the attack's failure are different, the model obtained from [59].

(b) *The attack's performance fluctuates with the small relative error caused by floating-point rounding errors.* When $0 \leq T < \lambda$, the computed gradient's relative error due to the floating-point rounding error is small and has the same variation pattern as the zero ratio and attack performance. The smallest relative error occurs around $T = 1$. However, it's crucial to note that at the bottom left of Figure 3 and Figure 4, the relative error and its detrimental effect on the gradient-based attack increase as T approaches zero after reaching the minimum.

It is straightforward to assume that since floating-point errors cause the above problem, it should be possible to solve the problem by simply increasing the precision of floating-point arithmetics because it can effectively reduce the floating-point errors. To test whether this simple idea works, we evaluated the impact of different floating-point arithmetics precisions, including half-precision, single-precision, and double-precision, on the model's robustness against FT-PGD with 100 iterations. As depicted in Figure 5, the attack's performance exhibits a similar variation pattern across different precisions, and the model's robustness measured with PGD remains nearly unchanged. While increasing the floating-point precision can delay attack failure due to floating-point underflow errors, it incurs significant computational overhead. Therefore, increasing the floating-point precision is not a viable solution to address the problem.

Surrogate loss functions(3,4) have been proposed [6, 10, 18] to work around floating-point underflow errors. They avoid suffering from floating-point underflow errors by discarding softmax operations. At the same time, they only use partial $\mathbf{z}$ elements in the new loss function. To demonstrate the effect of discarding elements of $\mathbf{z}$ on the attack's performance, We designed an experiment to make only a fixed number of $\mathbf{z}$ elements participate in the calculation of the gradients by exploiting floating-point underflow errors. To achieve this goal we scale the logits $\mathbf{z}_{\pi 1} - \mathbf{z}_{\pi i}$ by a scale factor c to make $c\,(\mathbf{z}_{\pi 1} - \mathbf{z}_{\pi i}) = \lambda$, such that $p^c_{\pi j} = 0$ if $j \geq i$ and $p^c_{\pi j} > 0$ if $j < i$, where $i \in \{2, 3, ..., K\}$ and $j \in \{1, 2, 3, ..., K\}$. So that only $i - 1$ number of $\mathbf{z}$ elements are involved in the gradient calculation, thus giving PGD the ability to attack the network with a fixed number of $\mathbf{z}$ elements, which we now call FNZ-PGD. We use FNZ-PGD with different numbers of $\mathbf{z}$ elements to evaluate the model's robustness which is overestimated mainly due to floating-point errors. From Figure 6, we can see that the more $\mathbf{z}$ elements involved in the attack, the more potent the attack will be. So discarding the $\mathbf{z}$ element in the loss function will impair the attack's performance when model robustness is overestimated mainly due to floating-point errors.

In summary, the relative error in the calculated gradients due to the floating-point errors, which include floating-point underflow errors and floating-point rounding errors, is the critical factor contributing to the overestimation of the model's robustness. While floating-point errors are a primary cause of gradient-based attacks' failure, they are not the sole cause. Reducing their impact can only address the portion of model robustness that is overestimated due to them. Therefore, it is unrealistic to expect that solely reducing the impact of floating-point errors alone will eliminate all problems related to overestimating model robustness.

Although we cannot eliminate the relative error in the gradients caused by floating-point errors, we can control it's side effect and significantly increase the power of gradient-based attacks. As a result, we propose the *Minimize the impact of floating-point errors* (MIFPE) loss, a small modification to the original CE loss, which scales the logits adaptively before the softmax operation. For untargeted and tar-

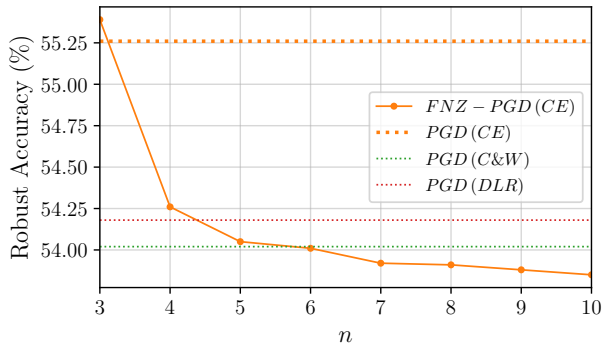


Figure 6. Using more $\mathbf{z}$ elements increases attack potency when model robustness is overestimated mainly due to floating-point errors. We utilized FNZ-PGD with 100 iterations, CE loss, and a baseline of PGD-100 with CE, C&W, and DLR loss under CIFAR10. All floating-point operations were single-precision, and the model was obtained from [59]. Here, n refers to the number of z elements used.

geted attacks, the targets are as follows respectively:

$$\mathcal{L}^{\text{MIFPE}}(\mathbf{z},y) \triangleq \mathcal{L}^{\text{ce}}(T\mathbf{z}/\Delta_{\text{detach}},y), \tag{7}$$

$$\mathcal{L}^{\text{MIFPE}}_{\text{target}}(\mathbf{z},y_t) = -\mathcal{L}^{\text{ce}}(T\mathbf{z}/\Delta_{\text{detach}},y_t), \tag{8}$$

Where y_t is a predefined class target, $t \in \{1,2,...,K\}$ and $y_t \neq y$. In our experiments, the factor T was set to around 1 to minimize the detrimental effect of floating-point errors on the gradient-based attacks.

The new surrogate loss $\mathcal{L}^{\text{MIFPE}}$ has fourfold advantages. First, it can eliminate the failure of gradient-based attacks due to floating-point underflow errors. Second, it can minimize the impact of the small relative error due to floating-point rounding errors and significantly improve the performance of gradient-based attacks. Third, it provides the flexibility to adjust the number of logit elements involved in the attack simply by changing the value of T. Finally, unlike the C&W or DLR loss, it still represents the original CE loss faithfully, and all values in the logit vector can still contribute to the gradient calculation.

4. Experiments

To ensure a fair comparison against other loss functions in our evaluation and to make our evaluation as comprehensive as possible, we conducted tests on different threat models and datasets. Specifically, we tested the ℓ_∞-norm bounded perturbation on the CIFAR-10 , CIFAR-100, MNIST [25] and ImageNet datasets [11],and the ℓ_2-norm threat model on the CIFAR-10 dataset. For each test, we ran the PGD with different loss functions, including the cross-entropy (CE) loss function, the C&W loss function [6], the DLR loss function [10], and our proposed MIFPE. We kept all other settings the same across the tests. Firstly, we used 100 iterations for each test. Secondly, we adopted the same step-size schedule with a linear decay [56], which updates the perturbation boundary ϵ at each iteration i according to the formula $2\epsilon(1-i/I)$, where I denotes the total number of iterations. Thirdly, we employed momentum-based updates [10] with a momentum factor of $\nu = 0.75$ for all tests. Lastly, we saved the best adversarial examples generated during the attack. To demonstrate the effectiveness of MIFPE in achieving near-state-of-the-art robustness with only 100 iterations, we compare our results with the best robust accuracy reported on RobustBench[1], which uses an ensemble of attacks and a minimum of 4900 iterations. We also compared the attack performance of MIFPE with GAMA_PGD [43] under the same 100-iteration constraint.

In Table 1, we present a comprehensive comparison showing that MIFPE outperforms CE, C&W, and DLR loss functions on all tested models. Additionally, we found that when PGD with momentum and step schedule using MIFPE as the loss function outperforms GAMA_PGD on all tested models by at leat more than 0.1% under 100 iterations constraint, which is impossible when PGD with momentum and step schedule using CE, C&W and DLR as the loss function. Remarkably, even with only 100 iterations, MIFPE achieves results that are very close to the best results obtained with at least 4900 iterations on some models, such as Uncovering limits [17] and Proper definition [32] for CIFAR10, $l_\infty, \varepsilon = 8/255$. Our findings also reveal that C&W and DLR perform even worse than CE in some cases, such as on Robustness library [14] and Fast adversarial training [51] for CIFAR10, l_∞, $\varepsilon = 8/255$. This suggests that discarding partial logit elements in the loss function would decrease the effectiveness of gradient-based attacks.

Faster convergence. We want to highlight that the effectiveness of MIFPE is not achieved by increasing the computational cost compared to other loss functions. We found that minimising the effect of the floating-point errors leads to faster convergence speed than competing loss functions. Figure 7 shows that for all two defending models, MIFPE loss performs more efficiently, and often with orders of magnitude faster than other loss functions for successful attacks. Finally, minimising the effect of floating-point error operation introduces no overhead and imposes no impact on the iteration time.

Boost the capability of existing attack strategies. Because MIFPE is a loss function, it can be easily integrated with existing attack strategies by replacing the original loss function. The results in Table 2 show that using MIFPE boosts the attack's effectiveness.

Ablation analysis. Our ablation experiments, presented in Table 3, investigated the effect of momentum and step-size schedule on PGD using different loss functions with 100 iterations. The results show that the MIFPE consis-

Table 1. Comparing the proposed MIFPE loss ($\mathcal{L}^{\text{MIFPE}}$), against CE ($\mathcal{L}^{\text{ce}}$), C&W ($\mathcal{L}^{\text{cw}}$), and DLR ($\mathcal{L}^{\text{dlr}}$) losses. For each surrogate loss, we use PGD-100 with the step-size schedule $2\epsilon\,(1 - i/I)$ and momentum $\nu = 0.75$. Models marked with † were trained with extra data. Models marked with ‡, which used $\varepsilon = 0.031$ as originally reported by the authors. Numbers in parentheses indicate the improvement w.r.t. the CE baseline. The best, calculated using an ensemble of attacks and a minimum of 4900 iterations, is reported from RobustBench[1] to demonstrate how closely MIFPE can approach the lowest known robustness accuracy with only 100 iterations.

Defense method	**Architecture**	**Clean**	**CE ($\mathcal{L}^{\text{ce}}$) 100**	**C&W ($\mathcal{L}^{\text{cw}}$) 100**		**DLR ($\mathcal{L}^{\text{dlr}}$) 100**		**GAMA_PGD 100**		**MIFPE ($\mathcal{L}^{\text{MIFPE}}$) 100**		**Best 4900**
MNIST, ℓ_∞, $\varepsilon = 0.3$												
Uncovering limits [17]	WRN-28-10	99.26	96.55	96.64	(+0.09)	96.71	(+0.16)	96.69	(+0.14)	**96.53**	**(−0.02)**	96.31
MMA training [12] †	LeNet5Madry	98.98	95.66	95.60	(-0.06)	95.56	(-0.10)	95.96	(+0.13)	**95.50**	**(−0.16)**	93.51
MMA training [12]	LeNet5Madry	98.95	95.09	95.33	(+0.24)	95.59	(+0.50)	95.74	(+0.19)	**94.88**	**(−0.21)**	91.40
Neural level sets [3]	SmallCNN	99.35	99.28	94.68	(-4.60)	95.09	(-4.19)	99.29	(+0.01)	**94.67**	**(−4.61)**	90.85
TRADES [59]	SmallCNN	99.48	93.69	93.88	(+0.19)	94.49	(+0.80)	93.82	(+0.13)	**93.67**	**(−0.02)**	92.71
Robust optimization [29]	SmallCNN	99.35	93.06	93.19	(+0.13)	93.63	(+0.57)	93.39	(+0.33)	**92.88**	**(−0.18)**	90.85
Fast adversarial training [51]	SmallCNN	98.50	86.82	86.96	(+0.14)	87.42	(+0.60)	87.62	(+0.80)	**86.57**	**(−0.25)**	82.93
CIFAR-10, ℓ_∞, $\varepsilon = 8/255$												
Uncovering limits [17]†	WRN-70-16	91.10	67.96	66.70	(-1.26)	66.78	(-1.18)	66.08	(-1.88)	**65.96**	**(−2.00)**	65.87
Fixing data augmentation [36]	WRN-106-16	88.50	67.57	65.55	(-2.02)	65.61	(-1.96)	64.94	(-2.63)	**64.75**	**(−2.82)**	64.58
Fixing data augmentation [36]	WRN-70-16	88.54	67.27	65.23	(-2.04)	65.32	(-1.95)	64.57	(-2.70)	**64.46**	**(−2.81)**	64.20
Proper definition [32]	WRN-70-16	89.01	66.66	63.94	(-2.72)	64.01	(-2.65)	63.65	(-3.01)	**63.49**	**(−3.17)**	63.35
Uncovering limits [17] †	WRN-28-10	89.48	65.59	63.62	(-1.97)	63.82	(-1.77)	63.05	(-2.90)	**62.96**	**(−2.63)**	62.76
Proper definition [32]	WRN-28-10	88.61	64.66	61.55	(-3.11)	61.62	(-3.04)	61.19	(-3.47)	**61.12**	**(−3.54)**	61.04
Adversarial weight perturbation [53]†	WRN-28-10	88.25	63.18	60.51	(-2.67)	60.60	(-2.58)	60.18	(-3.00)	**60.09**	**(−3.09)**	60.04
Unlabeled data [7]†	WRN-28-10	89.69	61.60	60.47	(-1.13)	60.67	(-0.93)	59.82	(-1.78)	**59.72**	**(−1.88)**	59.53
HYDRA [40]†	WRN-28-10	88.98	59.53	58.21	(-1.32)	58.30	(-1.23)	57.52	(-2.01)	**57.38**	**(−2.15)**	57.14
Misclassification-aware [50]	WRN-28-10	87.50	61.60	58.03	(-3.57)	58.73	(-2.87)	57.20	(-4.40)	**56.88**	**(−4.72)**	56.29
Pre-training [19]†	WRN-28-10	87.11	57.07	56.27	(-0.80)	57.07	(0.00)	55.22	(-1.85)	**55.10**	**(−1.97)**	54.92
Hypersphere embedding [33]	WRN-34-20	85.14	61.43	55.35	(-6.08)	56.21	(-5.22)	54.37	(-7.06)	**53.85**	**(−7.58)**	53.74
Overfitting [37]	WRN-34-20	85.34	56.85	55.22	(-1.63)	55.97	(-0.88)	53.87	(-2.98)	**53.62**	**(−3.23)**	53.42
Self-adaptive training [20]‡	WRN-34-10	83.48	56.12	54.30	(-1.82)	54.73	(-1.39)	53.64	(-2.48)	**53.48**	**(−2.64)**	53.34
TRADES [59]‡	WRN-34-10	84.92	55.21	53.94	(-1.27)	54.11	(-1.10)	53.38	(-1.83)	**53.22**	**(−1.99)**	53.08
Robustness library [14]	RN-50	87.03	51.56	52.07	(+0.51)	52.81	(+1.25)	50.04	(-1.52)	**49.84**	**(−1.72)**	49.25
Neural level sets [3]‡	RN-18	81.30	79.12	40.07	(-39.05)	45.10	(-34.02)	79.69	(+0.57)	**40.06**	**(−39.06)**	39.77
YOPO [58]	WRN-34-10	87.20	46.05	47.02	(+0.97)	47.55	(+1.50)	45.30	(-0.75)	**45.19**	**(−0.86)**	44.83
Fast adversarial training [51]	RN-18	83.34	45.75	45.81	(+0.06)	46.89	(+1.14)	43.71	(-2.04)	**43.57**	**(−2.18)**	43.21
CIFAR-100, ℓ_∞, $\epsilon = 8/255$												
Adversarial weight perturbation [54]	WRN-34-10	60.38	33.09	30.74	(-2.35)	31.13	(-1.96)	29.44	(-3.65)	**29.35**	**(−3.74)**	28.86
Pre-training [19]†	WRN-28-10	59.23	32.82	30.58	(-2.24)	31.83	(-0.99)	29.22	(-3.60)	**29.02**	**(−3.80)**	28.42
Progressive Hardening [42]	WRN-34-10	62.82	26.18	26.69	(+0.51)	27.26	(+1.08)	24.97	(-1.21)	**24.91**	**(−1.27)**	24.57
Overfitting [37]	RN-18	53.83	20.47	20.17	(-0.30)	20.30	(-0.17)	19.20	(-1.27)	**19.16**	**(−1.31)**	18.95
ImageNet, ℓ_∞, $\epsilon = 4/255$												
Transfer Better [39]	RN-50	64.02	38.44	37.26	(-1.18)	37.72	(-0.72)	34.94	(-3.50)	**34.84**	**(−3.60)**	34.96
Robustness library [14]	RN-50	62.56	32.16	32.24	(+0.08)	32.80	(+0.64)	29.72	(-2.44)	**29.60**	**(−2.56)**	29.22
Transfer Better [39]	RN-18	52.92	29.30	27.14	(-2.16)	27.40	(1.90)	25.56	(-3.74)	**25.40**	**(−3.90)**	18.95
CIFAR-10, ℓ_2, $\varepsilon = 0.5$												
Uncovering limits [17]†	WRN-70-16	94.74	81.71	80.93	(-0.78)	80.94	(-0.77)	87.74	(+6.03)	**80.57**	**(−1.14)**	80.53
Uncovering limits [17]	WRN-70-16	90.90	75.20	74.89	(-0.31)	74.95	(-0.25)	81.78	(+6.58)	**74.56**	**(−0.64)**	74.50
Adversarial weight perturbation [54]	WRN-34-10	88.51	74.77	73.88	(-0.89)	73.89	(-0.88)	79.38	(+4.61)	**73.67**	**(−1.10)**	73.66
Robustness library [14]	RN-50	90.83	69.65	70.13	(+0.48)	70.25	(+0.60)	78.15	(+8.50)	**69.29**	**(−0.36)**	69.24
Overfitting [37]	RN-18	88.67	68.66	68.74	(+0.08)	69.03	(+0.37)	76.19	(+7.53)	**67.87**	**(−0.79)**	67.68
Decoupling direction and norm [38]	WRN-28-10	89.05	66.56	67.00	(+0.44)	67.02	(+0.46)	74.61	(+8.05)	**66.48**	**(−0.08)**	66.44
MMA training [12]	WRN-28-4	88.02	66.22	66.58	(+0.36)	66.60	(+0.38)	71.44	(+5.22)	**66.16**	**(−0.06)**	66.09

tently outperforms the other loss functions, and shows the least variance across different setups. These findings suggest that the effectiveness of the MIFPE should not be dependent on the momentum and step-size schedule.

To analyze the reasons for the failure of the attack due to floating-point rounding errors, we selected $\mathbb{Z} = \mathbf{z}_y - \max_{i \neq y} \mathbf{z}_i$ because its sign indicates if the attack is successful or not. We compared the change of $\mathbb{Z}$ for samples attacked by PGD-100 with CE loss but failed, and PGD-100 with MIFPE loss and succeeded. Figure 8 demonstrates that for samples attacked with CE loss, the value of $\mathbb{Z}$ rapidly approaches 0 in the first few iterations and then remains positive afterwards. On the other hand, when attacked with MIFPE loss $\mathbb{Z}$ smoothly drops below 0 after approaching 0. This phenomenon suggests that the floating-point rounding error primarily may cause attack failure when the samples

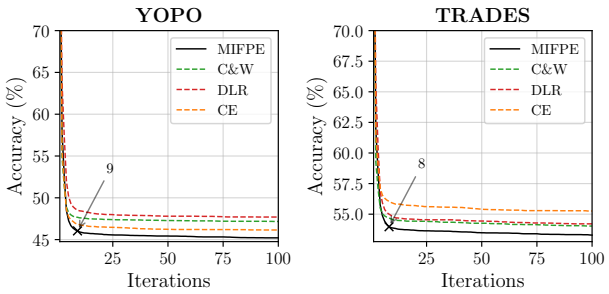


Figure 7. We compare MIFPE to standard CE loss and other surrogate loss functions (C&W [6] and DLR [10]) on defenders [58,59]. The graph shows iterations on the horizontal axis and the percentage of unsuccessful examples on the vertical axis.

Table 2. The comparison of attack performances for FGSM [16], PGD [29], APGD_DLR [10] and AA [10] using original loss functions versus MIFPE loss on CIFAR10 dataset and $\varepsilon = 0.031$ with the model obtained from TRADES [59]. with the ∇ row representing the improvement achieved with MIFPE.

Attack	FGSM	PGD	APGD_DLR	AA
iteration	1	100	100	4900
Original	79.83	79.79	45.90	40.22
MIFPE	**49.76**	**40.06**	**40.49**	**39.89**
∇	30.07	39.73	5.41	0.33

Table 3. Ablation study on the effect of momentum and step-size schedule on the PGD with different loss functions and under 100 iterations on CIFAR10 dataset with the model obtained from Neural level sets [3]. New components are added in consecutive rows.

Accuracy under	Loss Function			
attack(%)	CE	C&W	DLR	MIFPE
PGD	55.46	54.23	54.39	**53.42**
+Momentum	55.25	54.02	54.18	**53.23**
+Step-size schedule	55.21	53.94	54.11	**53.22**

become close to the classification boundary due to the relative error and its detrimental effect on the gradient-based attack would increase as T approaches zero after reaching the minimum.

5. Conclusion

This paper reveals that relative error in calculated gradient caused by floating-point errors is a fundamental reason why gradient-based attacks fail to accurately evaluate the model's robustness. To efficiently and accurately evaluate the model's robustness, we introduce MIFPE , a new loss function that minimises the impact of floating-point errors on gradient-based attacks. Compared with other loss functions, MIFPE enjoys higher accuracy and faster convergence speed. Moreover, we find several surprising observations related to adversarial attacks: (1)Discarding partial elements of logits in the loss will impair the attack's performance; (2)The floating-point rounding errors lead to the attack's failure by causing an increasing relative error as the samples get closer to the classification boundary; (3)Increasing the precision of floating-point arithmetic cannot solve the problem of overestimating the model's robustness caused by floating-point errors, etc. We hope that the above observations will help discover more efficient and reliable loss functions in the future. Finally, MIFPE is open source with reproducible results [2].

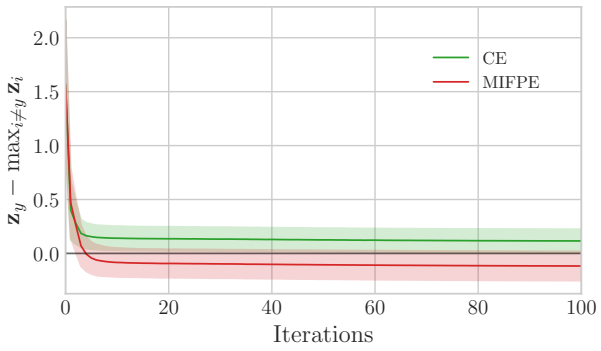


Figure 8. The changing process of the value of $\mathbb{Z} = \mathbf{z}_y - \max_{i \neq y} \mathbf{z}_i$ with the number of iterations during the attack on the CIFAR10 dataset using the model from TRADES [59] and single-precision arithmetic. The horizontal axes show the number of iterations used so far, and the vertical axes show the value of $\mathbb{Z}$.

Acknowledgments

This work is supported in part by Science and Technology Development Fund of Macao S.A.R (FDCT) under No. 0015/2019/AKP, 0123/2022/AFJ, and 0081/2022/A2. This work was carried out in part at SICC which is supported by SKL-IOTSC, University of Macau.

[2]Available at: https://github.com/MIFPE/Efficient-Loss-Function.

References

[1] Samet Akçay, Mikolaj E Kundegorski, Michael Devereux, and Toby P Breckon. Transfer learning using convolutional neural networks for object classification within X-ray baggage security imagery. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1057–1061. IEEE, 2016. 1

[2] Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are labels required for improving adversarial robustness? In *Advances in Neural Information Processing Systems*, volume 32, 2019. 1

[3] Matan Atzmon, Niv Haim, Lior Yariv, Ofer Israelov, Haggai Maron, and Yaron Lipman. Controlling neural level sets. In *Advances in Neural Information Processing Systems*, pages 2032–2041, 2019. 4, 7, 8

[4] Mariusz Bojarski, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Larry J Ackel, Urs Muller, Phil Yeres, and Karol Zieba. Visualbackprop: Efficient visualization of cnns for autonomous driving. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018. 1

[5] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019. 3

[6] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017. 1, 2, 5, 6, 8

[7] Yair Carmon, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems*, volume 32, pages 11192–11203, 2019. 7

[8] Taoran Cheng, Pengcheng Wen, and Yang Li. Research status of artificial neural network and its application assumption in aviation. In *2016 12th International Conference on Computational Intelligence and Security (CIS)*, pages 407–410. IEEE, 2016. 1

[9] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *ICML*, 2020. 2

[10] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020. 1, 2, 3, 5, 6, 8

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[12] Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Mma training: Direct input space margin maximization through adversarial training. In *International Conference on Learning Representations*, 2020. 7

[13] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li. Boosting adversarial attacks with momentum. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9185–9193, 2018. 1, 2, 3

[14] Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019. Available at: https://github.com/MadryLab/robustness. 6, 7

[15] Ian Goodfellow. Gradient masking causes clever to overestimate adversarial perturbation size. *arXiv preprint arXiv:1804.07870*, 2018. 1

[16] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 1, 2, 8

[17] Sven Gowal, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2020. 6, 7

[18] Sven Gowal, Jonathan Uesato, Chongli Qin, Po-Sen Huang, Timothy Mann, and Pushmeet Kohli. An alternative surrogate loss for PGD-based adversarial testing. *arXiv 1910.09338*, 2019. 2, 3, 5

[19] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721. PMLR, 2019. 7

[20] Lang Huang, Chao Zhang, and Hongyang Zhang. Self-adaptive training: beyond empirical risk minimization. In *NeurIPS*, 2020. 7

[21] Qian Huang, Isay Katsman, Horace He, Zeqi Gu, Serge Belongie, and Ser-Nam Lim. Enhancing adversarial example transferability with an intermediate level attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4733–4742, 2019. 3

[22] Nathan Inkawhich, Wei Wen, Hai Helen Li, and Yiran Chen. Feature space perturbations yield more transferable adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7066–7074, 2019. 3

[23] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *Technical Report, Google Inc.*, 2017. Available at: https://arxiv.org/abs/1607.02533. 2

[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015. 1

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 6

[26] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo R-CNN based 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7644–7652, 2019. 1

[27] Zhuoling Li, Minghui Dong, Shiping Wen, Xiang Hu, Pan Zhou, and Zhigang Zeng. CLU-CNNs: Object detection for medical images. *Neurocomputing*, 350:53–59, 2019. 1

[28] Ye Liu, Yaya Cheng, Lianli Gao, Xianglong Liu, Qilong Zhang, and Jingkuan Song. Practical evaluation of adversarial robustness via adaptive auto attack. 2022. 3

[29] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. 1, 2, 7, 8

[30] Xiaofeng Mao, Yuefeng Chen, Shuhui Wang, Hang Su, Yuan He, and Hui Xue. Composite adversarial attacks. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2021. 1

[31] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016. 1, 2

[32] Tianyu Pang, Min Lin, Xiao Yang, Jun Zhu, and Shuicheng Yan. Robustness and accuracy could be reconcilable by (Proper) definition. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17258–17277. PMLR, 17–23 Jul 2022. 6, 7

[33] Tianyu Pang, Xiao Yang, Yinpeng Dong, Kun Xu, Hang Su, and Jun Zhu. Boosting adversarial training with hypersphere embedding. In *NeurIPS*, 2020. 1, 3, 4, 7

[34] Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. In *Advances in Neural Information Processing Systems*, 2019. 3

[35] Ashish Kapoor Ratnesh Madaan. Game of drones at NeurIPS 2019: Simulation-based drone-racing competition built on AirSim. *Microsoft Research Blog*, 2019. Available at: https://www.microsoft.com/en-us/research/blog/game-of-drones-at-neurips-2019-simulation-based-drone-racing-competition-built-on-airsim/?OCID=msr_blog_gameofdrones_neurips_fb. 1

[36] Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy Mann. Data augmentation can improve robustness. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. 7

[37] Leslie Rice, Eric Wong, and J. Zico Kolter. Overfitting in adversarially robust deep learning. In *ICML*, 2020. 7

[38] Jérôme Rony, Luiz G Hafemann, Luiz S Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4322–4330, 2019. 7

[39] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems*, 33:3533–3545, 2020. 7

[40] Vikash Sehwag, Shiqi Wang, Prateek Mittal, and Suman Jana. Hydra: Pruning adversarially robust neural networks. *arXiv preprint arXiv:2002.10509*, 2020. 7

[41] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems*, volume 32, pages 3358–3369, 2019. 1

[42] Chawin Sitawarin, Supriyo Chakraborty, and David Wagner. Improving adversarial robustness through progressive hardening. *arXiv 2003.09347*, 2020. 7

[43] Gaurang Sriramanan, Sravanti Addepalli, Arya Baburaj, et al. Guided adversarial attack for evaluating and enhancing adversarial defenses. *Advances in Neural Information Processing Systems*, 33:20297–20308, 2020. 6

[44] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 2017. 2

[45] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. 1, 2

[46] Yusuke Tashiro, Yang Song, and Stefano Ermon. Diversity can be transferred: Output diversification for white- and black-box attacks. In *Advances in Neural Information Processing Systems*, 2020. 3

[47] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1

[48] Florian Tramer, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems*, 33:1633–1645, 2020. 3

[49] Guotai Wang, Wenqi Li, Maria A Zuluaga, Rosalind Pratt, Premal A Patel, Michael Aertsen, Tom Doel, Anna L David, Jan Deprest, Sébastien Ourselin, et al. Interactive medical image segmentation using deep learning with image-specific fine tuning. *IEEE transactions on medical imaging*, 37(7):1562–1573, 2018. 1

[50] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2020. 1, 7

[51] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020. 6, 7

[52] Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu. Does network width really help adversarial robustness? *arXiv 2010.01279*, 2020. 1

[53] Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu. Do wider neural networks really help adversarial robustness? In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 7

[54] Dongxian Wu, Shu tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. In *NeurIPS*, 2020. 1, 7

[55] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *IJCAI*, 2018. 1

[56] Yunrui Yu, Xitong Gao, and Cheng-Zhong Xu. LAFEAT: Piercing through adversarial defenses with latent features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5735–5745, June 2021. 3, 6

[57] Yunrui Yu, Xitong Gao, and Cheng zhong Xu. MORA: Improving ensemble robustness evaluation with model reweighing attack. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 3

[58] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. In *Advances in Neural Information Processing Systems*, volume 32, pages 227–238, 2019. 7, 8

[59] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019. 1, 4, 5, 6, 7, 8